QRS (QoS Routing Simulator)

Version 2.1

User's Manual [i]

Peng Zhang, Raimo Kantola

Networking Laboratory,

Helsinki University of Technology

Otakaari 5A, Espoo, Finland

Email: pgzhang@netlab.hut.fi, raimo.kantola@netlab.hut.fi

Tel: +358 9 4515454, +358 9 4512471

December, 2002

Keywords: Quality of Service (QoS), QoS Routing (QoSR), constraint based routing, routing protocols, performance analysis, modelling, discrete-event simulation, Resource Reservation (RSVP), Class-Based Queue (CBQ)

[i] This work is supported by IRoNet project carried out in Networking Laboratory, Helsinki University of Technology.

Content

1 Introduction 1
2 Obtaining, installing and starting QRS 1
2.1 Obtaining QRS 1
2.2 Installing QRS 1
2.3 Starting QRS 1
2.4 Recompiling QRS 2
3 User Interface 2
4 Network Configuration 3
4.1 Overview 3
4.2 Unchanged components 4
4.3 Extended and new components 5
4.3.1 Link 5
4.3.2 Realtime Traffic Source/Sink 6
4.3.3 RSVP 8
4.3.4 QOSPF 8
4.3.5 Resource Management 10
5 An Example 10
5.1 Configuring a network 11
5.2 Running the simulation 23
5.3 Analyzing the network 24

1 Introduction

QRS (QoS Routing Simulator) is designed to provide a flexible platform for the evaluation and comparison of QoS routing algorithms in IP networks. QRS allows users to define an arbitrary network configuration, control its simulation, log the values of selected parameters, and save, load, and modify the network configuration.

Acknowledgement

We are grateful to the users who help us in improving the performance of the simulator as well as writing the user's manual.

2 Obtaining, installing and starting QRS

2.1 Obtaining QRS

QRS is a public software that was developed at Helsinki University of Technology and is available at http://www.tct.hut.fi/tutkimus/ironet/QRS/index.html. The QRS version can be installed on Unix systems such as Solaris and Linux. Before installation, please make sure you download the software for the operating system you use.

The latest version is QRS v2.1. The filename of QRS version 2.1 is **qrs21.tar.gz**. It is stored as a compressed file that can be uncompressed by using gzip and restored by using tar.

2.2 Installing QRS

The installation steps are shown as follows.

(1) Setup a new directory, like /qrs and put the file qrs21.tar.gz into it.

(2) Uncompress the file by typing

gunzip -d qrs21.tar.gz

(3) Restore QRS by typing

tar xvf qrs21.tar

(4) Install QRS by typing

make

It is assumed that the user's operating system support gcc. The user may need to modify Makefile. If QRS can not be installed correctly, the user may ask the network administrator for help or contact us.

2.3 Starting QRS

After QRS is installed, it can be started by typing

qrs [options] [networkfile] [stoptime]

where networkfile, if present, contains the description of the target system. The stoptime value, if present, is used to stop the simulation when the simulated time exceeds this value. Its default unit is 10usec.

In the current version, the options are as follows.

-x

start the simulation of the target system immediately, without the X windows system interface. This option is compulsorily used in QRS.

-s seed

initialize the random number generator using the value of seed.

-i

When the simulated time exceeds the stoptime value specified, the user is asked whether or not to continue the simulation.

-record

write the occurrence of record type events to file sim_event.process_id.

-play filename

read and schedule events from the file filename, which should be a file that was created using the –record option.

-perfdt update-period

for periodically-updated performance measures use the value of update-period as the update period length in microseconds. The default unit of period is 10usec.

We give a few examples as follows. We assume the configuration file exists with the name tree.cfg.

(1). Run the example net and stop at 1 sec.

qrs –x example-net/tree.cfg 100000

(2). Run the example net, record log every 10msec, and stop at 1 sec.

qrs –x -perfdt 1000 example-net/tree.cfg 100000

2.4 Recompiling QRS

If any source file of QRS is modified, the user needs to recompile QRS by simply typing

make

in the installation directory. The user may need to modify Makefile.

3 User Interface

QRS only has a command line interface. The user configures the network by compiling the configuration file and looks up measurements by checking the recording files.

The user can define an arbitrary network configuration, control its simulation, log the values of selected parameters, and save, load, and modify network configurations.

Each instance of simulation produces two basic output files, i.e., sim_snap.ID and sim_log.ID. The former records the network configuration and the final value of all parameters. The latter file records the values of the parameters. They can be used together to analyze network statistics. Please see Chapter 5.2 for more explanation.

4 Network Configuration

4.1 Overview

QRS considers the target system as consisting of a physical network, a routing algorithm, a signalling, a resource management, and a workload, as illustrated in Figure 1.

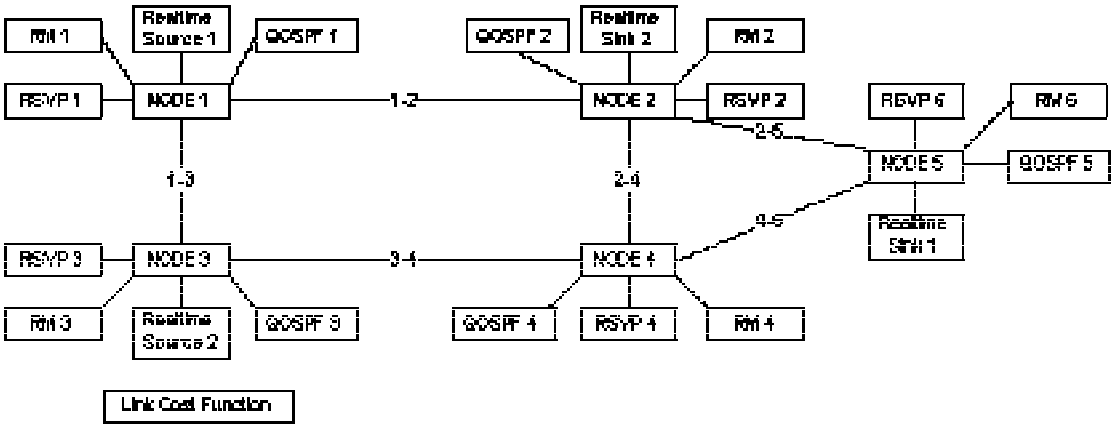

Figure 1. An example target system

The physical network consists of Link components and Node components. A Node component models the “physical” aspects of a store-and-forward entity. In the current version, it is characterized by parameters such as buffer space, processing speed, and failure and repair distribution. A Link component models a transmission channel between two nodes. In QRS, mechanism of class based queuing (CBQ) is also implemented in Link component. Therefore, a Link component is characterized by parameters of link (e.g., bandwidth, propagation delay and failure and repair distributions) and parameters of CBQ (e.g., EWMA-weight, EWMA-average). By connecting Link components and Node components, the user can specify a network of arbitrary topology.

The Routing component maintains at each node routing information that allows the node to route packets to their destinations. To realize a routing algorithm, a Routing component is connected to every node component. The Routing component contains the data structure representing the routing information and the functions to update these data structures and propagate their information. In QRS, only QOSPF is provided, which calculates routes based on two metrics, i.e., bandwidth and cost. In order to calculate QoS-based routes, two algorithms are provided: the lowest cost (LC) and the widest bandwidth (WB). Upon a request with bandwidth requirement, both algorithms first eliminate links whose available bandwidth is lower than bandwidth requirement and generate a new graph, then the LC algorithm selects the

path with the lowest cost while the WB algorithm selects the path with the widest available bandwidth in residual graph. Both algorithms can be realized based on Dijkstra algorithm or its variation.

In any case, the shortest path routing algorithm runs to compute the shortest path for BE traffic, i.e., default routing table. The user can explicitly choose one of these three algorithms for routing a traffic flow.

A Link Cost is needed to calculate the cost of each link. Four types of link cost functions are provided, based on hop count, utilization, delay, and hop-normalized delay. The user can choose one of the four functions by appropriately a Link-Cost component. There is exactly one link-cost component, and it is not connected to any other component. Link Cost in QRS is same as that in MaRS.

The Workload is defined in terms of source-sink pairs. Each source-sink pair consists of a Source component and a Sink component. These components are connected to node components (normally, the source's node component is different from the sink's node component). The source component produces packets and passes them to its node component. The packets are then forwarded by node components and link components until they reach their destination node, where they are consumed by sink component. A node component can have several Source/Sink components to it.

In QRS, the user can use four types of Source-Sink pairs: Realtime Traffic, FTP, TELNET, and Simple Traffic. Realtime Traffic is a new type of workload provided in QRS. The last three types are same as those in MaRS. Realtime Traffic has QoS requirement, i.e, bandwidth. Only Realtime Traffic submits QoS requests.

The Signalling is used to setup feasible path for Realtime Traffic. In QRS, RSVP is provided. RSVP component is connected to each Node component. The user can specify parameters of RSVP such as refreshing time.

The Resource Management (RM) is used to reserve resource (i.e., bandwidth) for paths.

There are two other components same as in MaRS, i.e., Performance Monitor and Stopper. The performance measures can be divided into two classes: periodically updated or event-updated. Periodically updated performance measures are updated periodically at time instants separated by a fixed length interval called the update period. Event-updated performance measures are updated whenever a related event occurs.

All the components mentioned above are provided in QRS. If these components are not enough for some specific purposes, the user can define new components, e.g., a new workload component or a new routing component. The user can get more details in [2] [3].

Currently, QRS can be invoked only in command line.

In the following subsections, the components are described for building configuration files. **Notably, we only present new parameters developed in QRS. All other parameters can be found in [1].**

4.2 Unchanged components

QRS imports some components from MaRS without change. These components include

- Node
- FTP
- Telnet
- Simple Traffic
- Cost Function
- Performance Monitor
- Stopper

4.3 Extended and new components

QRS extends LINK component and develops a few new components including

- Realtime Traffic: This type of traffic asks for path setup with bandwidth requirement. It is developed on the basis of Simple Traffic.
- RSVP: This is simplified version of RSVP and handles QoS requirements from Realtime traffic.
- QOSPF: This is a link-state based routing component. It implements several QoS routing algorithms.
- Resource management: This handles resource management, e.g., resource reservation.

4.3.1 Link

In addition to the parameters of Link in MaRS, a number of new parameters are developed in QRS in relation to CBQ, e.g., EWMA-weight, EWMA-average, and CBQ queue length, etc. The estimator of CBQ in QRS is EWMA (Exponential Weighted Moving Average) [4], which is also used in TCP to compute the average round trip delay. The formula is described as follows.

Upon coming a packet at time t, compute interval average at time t,

interval_average (t) = (1 – weight) * interval_average (t') + weight * (t-t')

, where weight is a constant and t' is the last time receiving a packet.

There are a number of traffic classes, e.g., EF, AF, and BE, resulting in six CBQ queue lengths (i.e., EF, AF1, AF2, AF3, AF4, BE) on each direction of a link. To each AF class, a WRED (Weighted Random Early Detection) with three precedence is provided. However, WRED is not fully implemented yet.

Input parameters:

- EWMA-weight

  The parameter is a constant used to compute average rate.

- EWMA-average

The parameter specifies the limit of interval average. If the interval average at time t is larger than the limit, specific traffic scheduling can be used based on CBQ principles [4].

- CBQ queue size

  The parameter specifies the queue size for a class.

- CBQ queue bandwidth

  The parameter specifies the queue bandwidth for a class. The total bandwidth of all queues should be equal or smaller than the link bandwidth. Note that there is an additional parameter for the link bandwidth.

- WRED threshold

  It is not used yet.

- WRED slop

  It is not used yet.

4.3.2 Realtime Traffic Source/Sink

Realtime Traffic has some new parameters except all same parameters as Simple Traffic.

Input parameters:

- Flow index

  Each pair of Realtime Traffic Source/Sink is numbered by the flow index number. In QRS v2.0, the flow index should be an integer within [0, 99]. Different pairs of Realtime Traffic must have different flow index numbers. In QRS v2.x, the *flow index* can also refer to the *label* used in MPLS.

- Class type

  The parameter corresponds to the class of the traffic, e.g., Class EF, AF and BE.

- Flow rate [bytes/sec]

  The parameter specifies the bandwidth requirement of the Realtime Traffic flow.

- Flow delay

  The parameter specifies the delay requirement of the Realtime Traffic flow. Currently, it is not used.

- Refreshment period

  The parameter specifies the refreshment period for RSVP PATH and RESV refreshment. The unit is 10usec.

- Starting time

  The parameter specifies the mean time between the simulator starts up and Realtime Traffic begins to request for past setup. The unit is 10usec.

- Distribution of starting time

The parameter specifies the distribution type of the starting time.

- Standard deviation of starting time

  The parameter specifies the standard deviation of the starting time.

- Flow Producing

  The parameter specifies the period that Realtime Traffic produces packets. The unit is 10usec.

- Flow Pausing

  The parameter specifies the period that Realtime Traffic pauses. The unit is 10usec. If this parameter is set to 0, the parameter of flow producing is ignored, that is, the Realtime traffic will not stop.

- Drop precedence

  The parameter specifies the drop precedence if the flow is an AF traffic. It is not used yet.

- Resource reserve type

  The value 0 stands for the type with resource reservation of this traffic. The value 1 stands for the type without resource reservation of this traffic. It is not used yet.

- Routing method

  This parameter specifies the routing method that is used for finding routes for this traffic. Value 0 stands for the lowest cost algorithm, value 1 stands for the widest bandwidth algorithm and 2 stands for the shortest path algorithm.

- Number of multi-paths

  This parameter specifies the number of multi-paths for this traffic. The default value is 1. The maximum 2 is considered in QRS v2.0. It is not fully supported yet.

- Load sharing method

  This parameter specifies the load sharing method that is used for sharing the traffic load over multipaths for this traffic. Only 0 is considered in QRS v2.0. It is not fully supported yet.

- Load sharing percent

  This parameter specifies the load sharing percent for the first path of the traffic. For example, if the number of multipaths is 1, the value should be 100. If the number of multipaths is 2, the value *w* could be within [1,100]. The load sharing percent of the other path is 100-*w*. It is not fully supported yet.

Output parameter:

- Flow table

  The flow table consists of TSpec and RSpec. It can be logged for maintenance.

4.3.3 RSVP

RSVP setups paths for forwarding data traffic of Realtime Traffic. RSVP performs path setup hop by hop. RSVP has no input parameters.

It has output parameters:

- Number of flows [bytes/sec]

  The parameter records the number of active flows in the flow table of RSVP at the node.

- Average RSVP packets

  The parameter records the average number of RSVP packets handled at the node.

- Total RSVP packets

  The parameter records the total number of RSVP packets handled at the node.

- Total PATH packets

  The parameter records the total number of RSVP PATH packets handled at the node.

- Total RESV packets

  The parameter records the total number of RSVP RESV packets handled at the node.

- Total PATH_ERR packets

  The parameter records the total number of RSVP PATH_ERR packets handled at the node.

- Total RESV_ERR packets

  The parameter records the total number of RSVP RESV_ERR packets handled at the node.

- Total PATH_TEAR packets

  The parameter records the total number of RSVP PATH_TEAR packets handled at the node.

- Total RESV_TEAR packets

  The parameter records the total number of RSVP RESV_TEAR packets handled at the node.

4.3.4 QOSPF

QOSPF has such input parameters:

- Time between topology broadcast [msec]

  The parameter specifies the mean value of interval time for broadcasting topology information. This parameter is neglected if the Period Based (PB) link state update algorithm is used.

- Standard deviation [msec]

The parameter specifies the stand deviation of the time between topology broadcast.

- Routing method

  The parameter specifies which route computation algorithm will be used. The value of 0 indicates the algorithm of Lowest Cost (**LC**). The value of 1 indicates the algorithm of Widest Bandwidth (**WB**).

- Link state update method

  The parameter specifies which link state update algorithm will be used. The value of 0 indicates Period Based algorithm (**PB**). The value of 1 indicates Threshold Based algorithm (**TB**). The value of 2 indicates Equal Class Based algorithm (**ECB**). Finally, the value of 3 indicates Unequal Class Based algorithm (**UCB**).

- LS update period

  The parameter specifies the LS update period if the algorithm of PB method is used. The unit is 10usec.

- Variation percent

  The parameter specifies the value of variation percent if the algorithm of TB method is used.

- Hold timer

  The parameter specifies the interval of LS advertising in case that the frequency of LS changing is too high. It is combined with the parameter of Variation percent. The unit is 10usec. A LS advertising is triggered only when the degree of LS changing is more than Variation percent and the difference between the current time and last LS advertising time is more than hold timer.

- Number of class

  The parameter specifies the number of classes if ECB method is used.

- Size_of_baseclass

  The parameter specifies the size of base class (B) if UCB update method is used.

- Factor_of_classsize

  The parameter specifies the value of factor (f) which is used to define unequal size classes: (0, B)(B,(f+1)B), ((f+1)B, (f^2+f+1)B, …if UCB update method is used.

Output parameters:

- Global topology table of cost

  It records the global link state database in term of cost.

- Global topology table of available bandwidth

  It records the global link state database in term of available bandwidth.

- Local topology table

  It records the local link state database.

- Routing table

  It records the routing table for BE traffic.

- Flow routing table

  The flow routing table consists of routes for Realtime traffic flows.

- Flow table

  It records the parameters of flows.

- Time Cost

  The parameter records the time consumed by the QOSPF in a node.

4.3.5 Resource Management

The component decides if a path can be established at a node. It simply follows such role:

If required bandwidth >= Link available bandwidth

Accept, then, reduce link available bandwidth and update local link state database.

Otherwise,

Reject the request.

The component has no input parameters. It has the following output parameters:

- Local topology table

  It records the local link state database for logging.

- Routing table

  It records the routing table for BE traffic for logging.

5 An Example

In this section, we give an example of defining a network configuration, controlling its simulation, logging the values of selected parameters. The example configuration file is also included in downloaded package. It resides in <qrs-install>/example-net directory.

5.1 Configuring a network

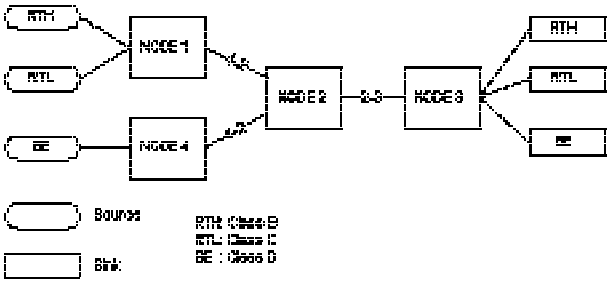

Figure 1 Tree Topology

The topology of the example network[ii] is shown in Figure 1.

```
# Tree topology
component 'node1' NODE 0 0 (see Note 1)
param 'node1' 32 0   # node1 (see Note 2)
param 1000 82 0      # Delay to process a packet (uSec): 1000
pflags 0 0     # Speed of node (uSec/kbyte): 0
param -1 82 0  # Buffer space in bytes (-1=inf): -1
param -1 82 0  # Mean time btw failures (sec): -1
param 0 82 0   # Interfailure dist (0=>EXP, 1=>UNIF): 0
param 0 82 0   # Enter standard deviation if UNIF: 0
param 1200 82 0      # Mean time to repair (sec): 1200
param 0 82 0   # Repair time dist (0=>EXP, 1=>UNIF): 0
param 0 82 0   # Enter standard deviation if UNIF: 0
pflags 26 0    # Node status: Up
pflags 6e 4    # Buffer space used: 0 (see Note 3)
pflags 6e 4    # Max buffer space used: 596
pflags 2e 4    # Number of packets dropped: 0
pflags 2e 4    # Instantaneous drop rate: 0
pflags 2e 4    # Memory utilization: 0
pflags 2e 4    # Input routing queue has 0 pkts
pflags 2a 8    # flow table
pflags 2e 4    # lk1-2 output queue has 0 pkts

component 'node2' NODE 0 0
param 'node2' 32 0   # node2
param 1000 82 0      # Delay to process a packet (uSec): 1000
pflags 0 0     # Speed of node (uSec/kbyte): 0
param -1 82 0  # Buffer space in bytes (-1=inf): -1
param -1 82 0  # Mean time btw failures (sec): -1
param 0 82 0   # Interfailure dist (0=>EXP, 1=>UNIF): 0
param 0 82 0   # Enter standard deviation if UNIF: 0
param 1200 82 0      # Mean time to repair (sec): 1200
param 0 82 0   # Repair time dist (0=>EXP, 1=>UNIF): 0
param 0 82 0   # Enter standard deviation if UNIF: 0
pflags 26 0    # Node status: Up
pflags 2e 4    # Buffer space used: 3200
pflags 2e 4    # Max buffer space used: 3884
pflags 2e 4    # Number of packets dropped: 0
pflags 2e 4    # Instantaneous drop rate: 0
pflags 2e 4    # Memory utilization: 3200
pflags 2e 4    # Input routing queue has 59 pkts
pflags 2a 8    # flow table
pflags 2e 4    # lk1-2 output queue has 0 pkts
pflags 2e 4    # lk2-3 output queue has 3 pkts
pflags 2e 4    # lk4-2 output queue has 0 pkts

component 'node3' NODE 0 0
param 'node3' 32 0   # node3
param 1000 82 0      # Delay to process a packet (uSec): 1000
```

[ii] The name of the corresponding configuration file is tree.cfg.

```
pflags 0 0    # Speed of node (uSec/kbyte): 0
param -1 82 0 # Buffer space in bytes (-1=inf): -1
param -1 82 0 # Mean time btw failures (sec): -1
param 0 82 0  # Interfailure dist (0=>EXP, 1=>UNIF): 0
param 0 82 0  # Enter standard deviation if UNIF: 0
param 1200 82 0      # Mean time to repair (sec): 1200
param 0 82 0  # Repair time dist (0=>EXP, 1=>UNIF): 0
param 0 82 0  # Enter standard deviation if UNIF: 0
pflags 26 0   # Node status: Up
pflags 2e 4   # Buffer space used: 0
pflags 2e 4   # Max buffer space used: 104
pflags 2e 4   # Number of packets dropped: 0
pflags 2e 4   # Instantaneous drop rate: 0
pflags 2e 4   # Memory utilization: 0
pflags 2e 4   # Input routing queue has 0 pkts
pflags 2a 8   # flow table
pflags 2e 4   # lk2-3 output queue has 0 pkts

component 'node4' NODE 0 0
param 'node4' 32 0   # node4
param 1000 82 0      # Delay to process a packet (uSec): 1000
pflags 0 0           # Speed of node (uSec/kbyte): 0
param -1 82 0        # Buffer space in bytes (-1=inf): -1
param -1 82 0        # Mean time btw failures (sec): -1
param 0 82 0         # Interfailure dist (0=>EXP, 1=>UNIF): 0
param 0 82 0         # Enter standard deviation if UNIF: 0
param 1200 82 0      # Mean time to repair (sec): 1200
param 0 82 0         # Repair time dist (0=>EXP, 1=>UNIF): 0
param 0 82 0         # Enter standard deviation if UNIF: 0
pflags 26 0   # Node status: Up
pflags 2e 4   # Buffer space used: 1116
pflags 2e 4   # Max buffer space used: 2176
pflags 2e 4   # Number of packets dropped: 1
pflags 2e 4   # Instantaneous drop rate: 0
pflags 2e 4   # Memory utilization: 1116
pflags 2e 4   # Input routing queue has 0 pkts
pflags 2a 8   # flow table
pflags 2e 4   # lk4-2 output queue has 3 pkts

component 'lk1-2' LINK 0 0
param 'lk1-2' 12 0   # lk1-2
param 1000 82 0      # Link propagation delay (USECS): 1000
param 750000 82 0    # Link bandwidth (bytes/sec): 750000
param 400 82 0       # Mean time btw failures (msecs): -1
param 1 82 0         # Interfailure dist (0=>EXP, 1=>UNIF): 1
param 0 82 0         # Enter standard deviation if UNIF: 0
param 300 82 0       # Mean time to repair (msecs): -1
param 1 82 0         # Repair time dist (0=>EXP, 1=>UNIF): 1
param 0 82 0         # Enter standard deviation if UNIF: 10
param 0.8 82 0       # CBQ: EWMA weight  : 0.8 (see Note 5)
param 150000 82 0    # CBQ: EWMA average limit  : 150000 (see Note 5)
param 50000 82 0     # CBQ1: class A queue size : 50000 (see Note 6)
param 50000 82 0     # CBQ1: class B queue size : 50000
param 50000 82 0     # CBQ1: class C queue size : 50000
param 50000 82 0     # CBQ1: class D queue size : 50000
param 187500 82 0    # CBQ1: Class A queue bandwidth : 187500
param 187500 82 0    # CBQ1: Class B queue bandwidth : 187500
param 187500 82 0    # CBQ1: Class C queue bandwidth : 187500
param 187500 82 0    # CBQ1: Class D queue bandwidth : 187500
param 50000 82 0     # CBQ2: class A queue size : 50000
param 50000 82 0     # CBQ2: class B queue size : 50000
param 50000 82 0     # CBQ2: class C queue size : 50000
param 50000 82 0     # CBQ2: class D queue size : 50000
param 187500 82 0    # CBQ2: Class A queue bandwidth : 187500
param 187500 82 0    # CBQ2: Class B queue bandwidth: 187500
param 187500 82 0    # CBQ2: Class C queue bandwidth: 187500
param 187500 82 0    # CBQ2: Class D queue bandwidth: 187500
param 50000 82 0     # CBQ1: BE queue size : 0
```

```
param 50000 82 0     # CBQ1: EF queue size : 0
param 187500 82 0    # CBQ1: BE queue bandwidth: 0
param 187500 82 0    # CBQ1: EF queue bandwidth: 0
param 1 82 0  # CBQ1: AF WRED threshold 1: 0 (see Note 15)
param 1 82 0  # CBQ1: AF WRED threshold 2: 0
param 1 82 0  # CBQ1: AF WRED threshold 3: 0
param 1 82 0  # CBQ1: AF WRED slop 1: 0 (see Note 16)
param 1 82 0  # CBQ1: AF WRED slop 2: 0
param 1 82 0  # CBQ1: AF WRED slop 3: 0
param 50000 82 0     # CBQ2: BE queue size : 0
param 50000 82 0     # CBQ2: EF queue size : 0
param 187500 82 0    # CBQ2: BE queue bandwidth: 0
param 187500 82 0    # CBQ2: EF queue bandwidth: 0
param 1 82 0  # CBQ2: AF WRED threshold 1: 0
param 1 82 0  # CBQ2: AF WRED threshold 2: 0
param 1 82 0  # CBQ2: AF WRED threshold 3: 0
param 1 82 0  # CBQ2: AF WRED slop 1: 0
param 1 82 0  # CBQ2: AF WRED slop 2: 0
param 1 82 0  # CBQ2: AF WRED slop 3: 0
pflags 26 0   # Link status: Up
pflags 6 0    # Failure status: 0
pflags 2e 2   # Inst. Data Util. , node1->node2: 0
pflags 2e 2   # Inst. Rout. Util., node1->node2: 0
pflags 2e 2   # Inst. Data Util. , node2->node1: 0
pflags 2e 2   # Inst. Rout. Util., node2->node1: 0

component 'lk2-3' LINK 0 0
param 'lk2-3' 12 0   # lk2-3
param 1000 82 0      # Link propagation delay (USECS): 1000
param 750000 82 0    # Link bandwidth (bytes/sec): 750000
param -1 82 0 # Mean time btw failures (msecs): 500
param 1 82 0  # Interfailure dist (0=>EXP, 1=>UNIF): 1
param 0 82 0  # Enter standard deviation if UNIF: 0
param -1 82 0 # Mean time to repair (msecs): 100
param 0 82 0  # Repair time dist (0=>EXP, 1=>UNIF): 0
param 0 82 0  # Enter standard deviation if UNIF: 0
param 0.8 82 0       # CBQ: EWMA weight  : 0.8
param 1.5e+06 82 0   # CBQ: EWMA average limit  : 1.5e+06
param 50000 82 0     # CBQ1: class A queue size : 50000
param 50000 82 0     # CBQ1: class B queue size : 50000
param 50000 82 0     # CBQ1: class C queue size : 50000
param 50000 82 0     # CBQ1: class D queue size : 50000
param 187500 82 0    # CBQ1: Class A queue bandwidth : 187500
param 187500 82 0    # CBQ1: Class B queue bandwidth : 187500
param 187500 82 0    # CBQ1: Class C queue bandwidth : 187500
param 187500 82 0    # CBQ1: Class D queue bandwidth : 187500
param 50000 82 0     # CBQ2: class A queue size : 50000
param 50000 82 0     # CBQ2: class B queue size : 50000
param 50000 82 0     # CBQ2: class C queue size : 50000
param 50000 82 0     # CBQ2: class D queue size : 50000
param 187500 82 0    # CBQ2: Class A queue bandwidth : 187500
param 187500 82 0    # CBQ2: Class B queue bandwidth: 187500
param 187500 82 0    # CBQ2: Class C queue bandwidth: 187500
param 187500 82 0    # CBQ2: Class D queue bandwidth: 187500
param 50000 82 0     # CBQ1: BE queue size : 0
param 50000 82 0     # CBQ1: EF queue size : 0
param 187500 82 0    # CBQ1: BE queue bandwidth: 0
param 187500 82 0    # CBQ1: EF queue bandwidth: 0
param 1 82 0  # CBQ1: AF WRED threshold 1: 0
param 1 82 0  # CBQ1: AF WRED threshold 2: 0
param 1 82 0  # CBQ1: AF WRED threshold 3: 0
param 1 82 0  # CBQ1: AF WRED slop 1: 0
param 1 82 0  # CBQ1: AF WRED slop 2: 0
param 1 82 0  # CBQ1: AF WRED slop 3: 0
param 50000 82 0     # CBQ2: BE queue size : 0
param 50000 82 0     # CBQ2: EF queue size : 0
param 187500 82 0    # CBQ2: BE queue bandwidth: 0
param 187500 82 0    # CBQ2: EF queue bandwidth: 0
```

```
param 1 82 0  # CBQ2: AF WRED threshold 1: 0
param 1 82 0  # CBQ2: AF WRED threshold 2: 0
param 1 82 0  # CBQ2: AF WRED threshold 3: 0
param 1 82 0  # CBQ2: AF WRED slop 1: 0
param 1 82 0  # CBQ2: AF WRED slop 2: 0
param 1 82 0  # CBQ2: AF WRED slop 3: 0
pflags 26 0   # Link status: Up
pflags 6 0    # Failure status: 0
pflags 2e 2   # Inst. Data Util. , node2->node3: 0
pflags 2e 2   # Inst. Rout. Util., node2->node3: 0
pflags 2e 2   # Inst. Data Util. , node3->node2: 0
pflags 2e 2   # Inst. Rout. Util., node3->node2: 0

component 'lk4-2' LINK 0 0
param 'lk4-2' 12 0   # lk4-2
param 1000 82 0      # Link propagation delay (USECS): 1000
param 750000 82 0    # Link bandwidth (bytes/sec): 750000
param -1 82 0 # Mean time btw failures (msecs): -1
param 0 82 0  # Interfailure dist (0=>EXP, 1=>UNIF): 0
param 0 82 0  # Enter standard deviation if UNIF: 0
param 10 82 0 # Mean time to repair (msecs): 10
param 0 82 0  # Repair time dist (0=>EXP, 1=>UNIF): 0
param 0 82 0  # Enter standard deviation if UNIF: 0
param 0.8 82 0       # CBQ: EWMA weight  : 0.8
param 150000 82 0    # CBQ: EWMA average limit  : 150000
param 50000 82 0     # CBQ1: class A queue size : 50000
param 50000 82 0     # CBQ1: class B queue size : 50000
param 50000 82 0     # CBQ1: class C queue size : 50000
param 50000 82 0     # CBQ1: class D queue size : 50000
param 187500 82 0    # CBQ1: Class A queue bandwidth : 187500
param 187500 82 0    # CBQ1: Class B queue bandwidth : 187500
param 187500 82 0    # CBQ1: Class C queue bandwidth : 187500
param 187500 82 0    # CBQ1: Class D queue bandwidth : 187500
param 50000 82 0     # CBQ2: class A queue size : 50000
param 50000 82 0     # CBQ2: class B queue size : 50000
param 50000 82 0     # CBQ2: class C queue size : 50000
param 50000 82 0     # CBQ2: class D queue size : 50000
param 187500 82 0    # CBQ2: Class A queue bandwidth : 187500
param 187500 82 0    # CBQ2: Class B queue bandwidth: 187500
param 187500 82 0    # CBQ2: Class C queue bandwidth: 187500
param 187500 82 0    # CBQ2: Class D queue bandwidth: 187500
param 50000 82 0     # CBQ1: BE queue size : 0
param 50000 82 0     # CBQ1: EF queue size : 0
param 187500 82 0    # CBQ1: BE queue bandwidth: 0
param 187500 82 0    # CBQ1: EF queue bandwidth: 0
param 1 82 0  # CBQ1: AF WRED threshold 1: 0
param 1 82 0  # CBQ1: AF WRED threshold 2: 0
param 1 82 0  # CBQ1: AF WRED threshold 3: 0
param 1 82 0  # CBQ1: AF WRED slop 1: 0
param 1 82 0  # CBQ1: AF WRED slop 2: 0
param 1 82 0  # CBQ1: AF WRED slop 3: 0
param 50000 82 0     # CBQ2: BE queue size : 0
param 50000 82 0     # CBQ2: EF queue size : 0
param 187500 82 0    # CBQ2: BE queue bandwidth: 0
param 187500 82 0    # CBQ2: EF queue bandwidth: 0
param 1 82 0  # CBQ2: AF WRED threshold 1: 0
param 1 82 0  # CBQ2: AF WRED threshold 2: 0
param 1 82 0  # CBQ2: AF WRED threshold 3: 0
param 1 82 0  # CBQ2: AF WRED slop 1: 0
param 1 82 0  # CBQ2: AF WRED slop 2: 0
param 1 82 0  # CBQ2: AF WRED slop 3: 0
pflags 26 0   # Link status: Up
pflags 6 0    # Failure status: 0
pflags 2e 2   # Inst. Data Util. , node4->node2: 0
pflags 2e 2   # Inst. Rout. Util., node4->node2: 0
pflags 2e 2   # Inst. Data Util. , node2->node4: 0
pflags 2e 2   # Inst. Rout. Util., node2->node4: 0
```

```
component 'realtime_node1-node3_so_1' REALTIME_SOURCE 0 0
param 'realtime_node1-node3_so_1' 12 0   # realtime_node1-node3_so_1
pflags 6 0     # Peer: node3.realtime_node1-node3_si_1
param 0 82 0   # Not select(0), select(1) : 0
param 512 82 0       # Ave Packet length: 512
param 1000 82 0  # RealtimeTraffic starting to request: 0
param 1 82 0  # Choose dist (0=>EXP, 1=>UNIF, 2=>CBR: 0
param 0 82 0  # Enter standard deviation if UNIF or FBN: 0
param 10000 82 0 # Enter the interval(us) of traffic produce: 10000
param 5000 82 0  # Enter the interval(us) of traffic pause: 5000
param 2200 82 0      # Ave delay btw packets (uSec): 2200
param 0 82 0   # Choose dist (0=>EXP, 1=>UNIF, 2=>CBR, 3=: 0
param 0 82 0   # Enter standard deviation if UNIF or FBN: 0
param -1 82 4  # Window size (-1 for inf): -1
pflags 2e 4    # Packets Sent: 24
pflags 2e 4    # Packets Acked: 21
pflags 2e 4    # Packets Received: 0
pflags 2e 4    # Packets Lost: 0
param 0 82 0   # Enter Hurst parameter if FBN: 0
param 1 82 0   # Flow index of this source : 1 (see Note 7)
param 1 82 0   # Class type of this source : 1 (see Note 8)
param 0 86 0   # Drop precedence of this source : 0 (see Note 17)
param 0 86 0   # Resource reserve type of this source: WR: 0 (see Note 18)
param 0 86 0   # Routing method for this source : 1 (see Note 19)
param 1 86 0   # The number of multipaths: 1 or 2 : 0 (see Note 20)
param 0 82 0   # Load sharing method for this source: 0: : 0 (see Note 21)
param 100 82 0 # Load sharing percent, e.g., 10 (percent): 0 (see Note 22)
param 250000 86 0    # Flow rate of this source (bytes/sec) : 250000
param 0 86 0   # Flow delay of this source : 0
param 20000 86 0     # Refresh Period of this source (10us): 20000
pflags 2a 8    # Flow table

component 'realtime_node1-node3_si_1' REALTIME_SINK 0 0
param 'realtime_node1-node3_si_1' 12 0   # realtime_node1-node3_si_1
pflags 6 0     # Peer: node1.realtime_node1-node3_so_1
pflags 82 0    # Not select(0), select(1) : 0
pflags 82 0    # Ave Packet length: 512
pflags 82 0    # Ave delay btw packets (uSec): 2000
pflags 82 0    # Choose dist (0=>EXP, 1=>UNIF, 2=>CBR, 3=: 0
pflags 82 0    # Enter standard deviation if UNIF or FBN: 0
pflags 82 4    # Window size (-1 for inf): 20
pflags 2e 4    # Packets Sent: 0
pflags 2e 4    # Packets Acked: 0
pflags 2e 4    # Packets Received: 21
pflags 2e 4    # Ack Packets Lost: 0
pflags 82 0    # Enter Hurst parameter if FBN: 0
param 1 82 0   # Flow index of this sink : 1
param 1 82 0   # Class type of this sink : 1
param 0 86 0   # Drop precedence of this source : 0
param 0 86 0   # Resource reserve type of this source: WR: 0
param 0 86 0   # Routing method for this source : 1
param 1 86 0   # The number of multipaths: 1 or 2 : 0
param 0 82 0   # Load sharing method for this source: 0: : 0
param 100 82 0       # Load sharing percent, e.g., 10 (percent): 0
param 250000 86 0    # Flow rate of this sink : 250000
param 0 86 0   # Flow delay of this sink : 0
param 20000 86 0     # Refresh Period of this sink (10us): 20000
pflags 2a 8    # Flow table

component 'realtime_node1-node3_so_2' REALTIME_SOURCE 0 0
param 'realtime_node1-node3_so_2' 12 0   # realtime_node1-node3_so_2
pflags 6 0     # Peer: node3.realtime_node1-node3_si_2
param 0 82 0   # Not select(0), select(1) : 0
param 512 82 0       # Ave Packet length: 512
param 2000 82 0  # RealtimeTraffic starting to request: 0
param 1 82 0  # Choose dist (0=>EXP, 1=>UNIF, 2=>CBR: 0
param 0 82 0  # Enter standard deviation if UNIF or FBN: 0
param 15000 82 0 # Enter the interval(us) of traffic produce: 10000
```

```
param 6000 82 0  # Enter the interval(us) of traffic pause: 5000
param 1450 82 0      # Ave delay btw packets (uSec): 1450
param 0 82 0  # Choose dist (0=>EXP, 1=>UNIF, 2=>CBR, 3=: 0
param 0 82 0  # Enter standard deviation if UNIF or FBN: 0
param -1 82 4 # Window size (-1 for inf): -1
pflags 2e 4   # Packets Sent: 32
pflags 2e 4   # Packets Acked: 30
pflags 2e 4   # Packets Received: 0
pflags 2e 4   # Packets Lost: 0
param 0 82 0  # Enter Hurst parameter if FBN: 0
param 2 82 0  # Flow index of this source : 2
param 2 82 0  # Class type of this source : 2
param 0 86 0  # Drop precedence of this source : 0
param 0 86 0  # Resource reserve type of this source: WR: 0
param 0 86 0  # Routing method for this source : 1
param 1 86 0  # The number of multipaths: 1 or 2 : 0
param 0 82 0  # Load sharing method for this source: 0: : 0
param 100 82 0       # Load sharing percent, e.g., 10 (percent): 0
param 375000 86 0    # Flow rate of this source : 375000
param 0 86 0  # Flow delay of this source : 0
param 20000 86 0     # Refresh Period of this source (10us): 20000
pflags 2a 8   # Flow table

component 'realtime_node1-node3_si_2' REALTIME_SINK 0 0
param 'realtime_node1-node3_si_2' 12 0   # realtime_node1-node3_si_2
pflags 6 0    # Peer: node1.realtime_node1-node3_so_2
pflags 82 0   # Not select(0), select(1) : 0
pflags 82 0   # Ave Packet length: 512
pflags 82 0   # Ave delay btw packets (uSec): 2000
pflags 82 0   # Choose dist (0=>EXP, 1=>UNIF, 2=>CBR, 3=: 0
pflags 82 0   # Enter standard deviation if UNIF or FBN: 0
pflags 82 4   # Window size (-1 for inf): 20
pflags 2e 4   # Packets Sent: 0
pflags 2e 4   # Packets Acked: 0
pflags 2e 4   # Packets Received: 30
pflags 2e 4   # Ack Packets Lost: 0
pflags 82 0   # Enter Hurst parameter if FBN: 0
param 2 82 0  # Flow index of this sink : 2
param 2 82 0  # Class type of this sink : 2
param 0 86 0  # Drop precedence of this source : 0
param 0 86 0  # Resource reserve type of this source: WR: 0
param 0 86 0  # Routing method for this source : 1
param 1 86 0  # The number of multipaths: 1 or 2 : 0
param 0 82 0  # Load sharing method for this source: 0: : 0
param 100 82 0       # Load sharing percent, e.g., 10 (percent): 0
param 375000 86 0    # Flow rate of this sink : 375000
param 0 86 0  # Flow delay of this sink : 0
param 20000 86 0     # Refresh Period of this sink (10us): 20000
pflags 2a 8   # Flow table

component 'simple_node4-node3_so_1' SIMPLE_SOURCE 0 0
param 'simple_node4-node3_so_1' 12 0     # simple_node4-node3_so_1
pflags 6 0    # Peer: node3.simple_node4-node3_si_1
param 0 82 0  # Not select(0), select(1) : 0
param 512 82 0       # Ave Packet length: 512
param 2200 82 0      # Ave delay btw packets (uSec): 2200
param 0 82 0  # Choose dist (0=>EXP, 1=>UNIF): 0
param 0 82 0  # Enter standard deviation if UNIF: 0
param -1 82 4 # Window size (-1 for inf): -1
pflags 2e 4   # Packets Sent: 56
pflags 2e 4   # Packets Acked: 35
pflags 2e 4   # Packets Received: 0
pflags 2e 4   # Packets Lost: 1

component 'simple_node4-node3_si_1' SIMPLE_SINK 0 0
param 'simple_node4-node3_si_1' 12 0     # simple_node4-node3_si_1
pflags 6 0    # Peer: node4.simple_node4-node3_so_1
pflags 0 0    # Not select(0), select(1) : 0
```

```
pflags 0 0    # Ave Packet length: 512
pflags 0 0    # Ave delay btw packets (uSec): 2000
pflags 0 0    # Choose dist (0=>EXP, 1=>UNIF): 0
pflags 0 0    # Enter standard deviation if UNIF: 0
pflags 0 4    # Window size (-1 for inf): 20
pflags 4 4    # Packets Sent: 0
pflags 4 4    # Packets Acked: 0
pflags 2e 4   # Packets Received: 35
pflags 4 4    # Ack Packets Lost: 0

component 'qospf1' QOSPF 0 0
param 'qospf1' 12 0  # qospf1
param 100000 82 0    # Time btw topology broad (msec): 1000 (see Note 9)
param 10 82 0 # Standard deviation: 100
param 0 82 0  # Routing method: 0-> lowest cost, 1-> widest bandwidth
param 1 82 0  # LS update method: 0->PB, 1->TB, 2->ECB, 3->UCB
param 1000 82 0 #  value of LS update period (*10us): 1000 (see Note 10)
param 0.1 82 0  # update by bandwidth percent variation: 0.1 (see Note 11)
param 0 82 0  # value of hold timer: 0
param 21 82 0 # the number of classes for equal class up: 21
param 0.01 82 0      # the size ( >= 0.0000001 )of base class B: 0.01
param 2 82 0  # for unequal class update, f>1, (0,B),(B,: 2
pflags 2e 8   # total time cost in QOSPF: 0
pflags 4 4    # Sequence number: 11
pflags 2e 8   # Global topology table of Cost
pflags 2e 8   # Flow Global topology table of bandwidth
pflags 2e 8   # Local topology table
pflags 2e 8   # Routing table
pflags 2e 8   # Flow Routing table
pflags 4 0    # Last sequence no table
pflags 2e 8   # Flow table

component 'qospf2' QOSPF 0 0
param 'qospf2' 12 0  # qospf2
param 100000 82 0    # Time btw topology broad (msec): 1000
param 10 82 0 # Standard deviation: 200
param 0 82 0  # Routing method: 0-> lowest cost, 1-> wid: 0
param 1 82 0  # Link state update method: 0->PB, 1->TB, 2->ECB, 3->UCB
param 10000 82 0     #  value of LS update period : 1000
param 0.1 82 0       # update by bandwidth percent variation: 0.8
param 0 82 0  # value of hold timer: 0
param 21 82 0 # the number of classes for equal class up: 21
param 0.01 82 0      # the size ( >= 0.0000001 )of base class B: 0.01
param 2 82 0  # for unequal class update, f>1, (0,B),(B,: 2
pflags 2e 8   # total time cost in QOSPF: 0
pflags 4 4    # Sequence number: 11
pflags 2e 8   # Global topology table of Cost
pflags 2e 8   # Flow Global topology table of bandwidth
pflags 2e 8   # Local topology table
pflags 2e 8   # Routing table
pflags 2e 8   # Flow Routing table
pflags 4 0    # Last sequence no table
pflags 2e 8   # Flow table

component 'qospf3' QOSPF 0 0
param 'qospf3' 12 0  # qospf3
param 100000 82 0    # Time btw topology broad (msec): 1000
param 10 82 0 # Standard deviation: 200
param 0 82 0  # Routing method: 0-> lowest cost, 1-> wid: 0
param 1 82 0  # Link state update method: 0->PB, 1->TB, 2->ECB, 3->UCB
param 10000 82 0     #  value of LS update period : 1000
param 0.1 82 0       # update by bandwidth percent variation: 0.8
param 0 82 0  # value of hold timer: 0
param 21 82 0 # the number of classes for equal class up: 21
param 0.01 82 0      # the size ( >= 0.0000001 )of base class B: 0.01
param 2 82 0  # for unequal class update, f>1, (0,B),(B,: 2
pflags 2e 8   # total time cost in QOSPF: 0
pflags 4 4    # Sequence number: 11
```

```
pflags 2e 8   # Global topology table of Cost
pflags 2e 8   # Flow Global topology table of bandwidth
pflags 2e 8   # Local topology table
pflags 2e 8   # Routing table
pflags 2e 8   # Flow Routing table
pflags 4 0    # Last sequence no table
pflags 2e 8   # Flow table

component 'qospf4' QOSPF 0 0
param 'qospf4' 12 0  # qospf4
param 100000 82 0    # Time btw topology broad (msec): 1000
param 10 82 0 # Standard deviation: 200
param 0 82 0  # Routing method: 0-> lowest cost, 1-> wid: 0
param 1 82 0  # Link state update method: 0->PB, 1->TB, 2->ECB, 3->UCB
param 10000 82 0     #  value of LS update period : 1000
param 0.1 82 0       # update by bandwidth percent variation: 0.8
param 0 82 0  # value of hold timer: 0
param 21 82 0 # the number of classes for equal class up: 21
param 0.01 82 0      # the size ( >= 0.0000001 )of base class B: 0.01
param 2 82 0  # for unequal class update, f>1, (0,B),(B,: 2
pflags 2e 8   # total time cost in QOSPF: 0
pflags 4 4    # Sequence number: 11
pflags 2e 8   # Global topology table of Cost
pflags 2e 8   # Flow Global topology table of bandwidth
pflags 2e 8   # Local topology table
pflags 2e 8   # Routing table
pflags 2e 8   # Flow Routing table
pflags 4 0    # Last sequence no table
pflags 2e 8   # Flow table

component 'rsvp1' RSVP 0 0
param 'rsvp1' 12 0   # rsvp1
pflags 2a 0   # Current flows in use: 0
pflags 2a 0   # Average RSVP packets per time interval: 0
pflags 2a 0   # Total RSVP packets: 0
pflags 2a 0   # Total RSVP PATH packets: 0
pflags 2a 0   # Total RSVP RESV packets: 0
pflags 2a 0   # Total RSVP PATH_ERR packets: 0
pflags 2a 0   # Total RSVP RESV_ERR packets: 0
pflags 2a 0   # Total RSVP PATH_TEAR packets: 0
pflags 2a 0   # Total RSVP RESV_TEAR packets: 0
pflags 2a 0   # Average RSVP packets per time interval: 0

component 'rsvp2' RSVP 0 0
param 'rsvp2' 12 0   # rsvp2
pflags 2a 0   # Current flows in use: 0
pflags 2a 0   # Average RSVP packets per time interval: 0
pflags 2a 0   # Total RSVP packets: 0
pflags 2a 0   # Total RSVP PATH packets: 0
pflags 2a 0   # Total RSVP RESV packets: 0
pflags 2a 0   # Total RSVP PATH_ERR packets: 0
pflags 2a 0   # Total RSVP RESV_ERR packets: 0
pflags 2a 0   # Total RSVP PATH_TEAR packets: 0
pflags 2a 0   # Total RSVP RESV_TEAR packets: 0
pflags 2a 0   # Average RSVP packets per time interval: 0

component 'rsvp3' RSVP 0 0
param 'rsvp3' 12 0   # rsvp3
pflags 2a 0   # Current flows in use: 0
pflags 2a 0   # Average RSVP packets per time interval: 0
pflags 2a 0   # Total RSVP packets: 0
pflags 2a 0   # Total RSVP PATH packets: 0
pflags 2a 0   # Total RSVP RESV packets: 0
pflags 2a 0   # Total RSVP PATH_ERR packets: 0
pflags 2a 0   # Total RSVP RESV_ERR packets: 0
pflags 2a 0   # Total RSVP PATH_TEAR packets: 0
pflags 2a 0   # Total RSVP RESV_TEAR packets: 0
pflags 2a 0   # Average RSVP packets per time interval: 0
```

```
component 'rsvp4' RSVP 0 0
param 'rsvp4' 12 0   # rsvp4
pflags 2a 0   # Current flows in use: 0
pflags 2a 0   # Average RSVP packets per time interval: 0
pflags 2a 0   # Total RSVP packets: 0
pflags 2a 0   # Total RSVP PATH packets: 0
pflags 2a 0   # Total RSVP RESV packets: 0
pflags 2a 0   # Total RSVP PATH_ERR packets: 0
pflags 2a 0   # Total RSVP RESV_ERR packets: 0
pflags 2a 0   # Total RSVP PATH_TEAR packets: 0
pflags 2a 0   # Total RSVP RESV_TEAR packets: 0
pflags 2a 0   # Average RSVP packets per time interval: 0

component 'rm1' RM 0 0
param 'rm1' 12 0     # rm1
pflags 2a 8   # Local topology table
pflags 2a 8   # flow table

component 'rm2' RM 0 0
param 'rm2' 12 0     # rm2
pflags 2a 8   # Local topology table
pflags 2a 8   # flow table

component 'rm3' RM 0 0
param 'rm3' 12 0     # rm3
pflags 2a 8   # Local topology table
pflags 2a 8   # flow table

component 'rm4' RM 0 0
param 'rm4' 12 0     # rm4
pflags 2a 8   # Local topology table
pflags 2a 8   # flow table

component 'CostFcn' LINK_COST_FUNC 0 0
pflags 2 0    # CostFcn
param 1 82 0  # Cost fcn 1:hndl 2:dly 3:util 4:hop: 1
param 100000 82 0    # Delay cost fcn max delay: 100000
param 10 82 0 # Delay cost fcn min delay: 10
param 10 82 0 # Slope: 10
param 0 82 0  # Offset: 0
param 2 82 0  # Cost movement limit: 2
param 10 82 0 # Maximum Cost: 10
param 1 82 0  # Minimum Cost: 1
param 0.5 82 0       # Exp filter, coeff of new: 0.5

neighbor1 'node1' 'lk1-2'  (see Note 12, 13)
neighbor1 'node1' 'realtime_node1-node3_so_1'
neighbor1 'node1' 'realtime_node1-node3_so_2'
neighbor1 'node1' 'qospf1'
neighbor1 'node1' 'rsvp1'
neighbor1 'node1' 'rm1'
neighbor1 'node2' 'lk1-2' 2e 4
neighbor1 'node2' 'lk2-3' 2e 4
neighbor1 'node2' 'lk4-2' 2e 4
neighbor1 'node2' 'qospf2'
neighbor1 'node2' 'rsvp2'
neighbor1 'node2' 'rm2'
neighbor1 'node3' 'lk2-3' 2e 4
neighbor1 'node3' 'realtime_node1-node3_si_1'
neighbor1 'node3' 'realtime_node1-node3_si_2'
neighbor1 'node3' 'simple_node4-node3_si_1'
neighbor1 'node3' 'qospf3'
neighbor1 'node3' 'rsvp3'
neighbor1 'node3' 'rm3'
neighbor1 'node4' 'lk4-2' 2e 4
neighbor1 'node4' 'simple_node4-node3_so_1'
neighbor1 'node4' 'qospf4'
```

```
neighbor1 'node4' 'rsvp4'
neighbor1 'node4' 'rm4'
neighbor1 'lk1-2' 'node1'
neighbor1 'lk1-2' 'node2'
neighbor1 'lk2-3' 'node2'
neighbor1 'lk2-3' 'node3'
neighbor1 'lk4-2' 'node4'
neighbor1 'lk4-2' 'node2'
neighbor1 'realtime_node1-node3_so_1' 'node1'
neighbor1 'realtime_node1-node3_so_1' 'rsvp1'
neighbor1 'realtime_node1-node3_si_1' 'node3'
neighbor1 'realtime_node1-node3_si_1' 'rsvp3'
neighbor1 'realtime_node1-node3_so_2' 'node1'
neighbor1 'realtime_node1-node3_so_2' 'rsvp1'
neighbor1 'realtime_node1-node3_si_2' 'node3'
neighbor1 'realtime_node1-node3_si_2' 'rsvp3'
neighbor1 'simple_node4-node3_so_1' 'node4'
neighbor1 'simple_node4-node3_si_1' 'node3'
neighbor1 'qospf1' 'node1'
neighbor1 'qospf1' 'rsvp1'
neighbor1 'qospf1' 'rm1'
neighbor1 'qospf2' 'node2'
neighbor1 'qospf2' 'rsvp2'
neighbor1 'qospf2' 'rm2'
neighbor1 'qospf3' 'node3'
neighbor1 'qospf3' 'rsvp3'
neighbor1 'qospf3' 'rm3'
neighbor1 'qospf4' 'node4'
neighbor1 'qospf4' 'rsvp4'
neighbor1 'qospf4' 'rm4'
neighbor1 'rsvp1' 'node1'
neighbor1 'rsvp1' 'realtime_node1-node3_so_1'
neighbor1 'rsvp1' 'realtime_node1-node3_so_2'
neighbor1 'rsvp1' 'qospf1'
neighbor1 'rsvp1' 'rm1'
neighbor1 'rsvp2' 'node2'
neighbor1 'rsvp2' 'qospf2'
neighbor1 'rsvp2' 'rm2'
neighbor1 'rsvp3' 'node3'
neighbor1 'rsvp3' 'realtime_node1-node3_si_1'
neighbor1 'rsvp3' 'realtime_node1-node3_si_2'
neighbor1 'rsvp3' 'qospf3'
neighbor1 'rsvp3' 'rm3'
neighbor1 'rsvp4' 'node4'
neighbor1 'rsvp4' 'qospf4'
neighbor1 'rsvp4' 'rm4'
neighbor1 'rm1' 'node1'
neighbor1 'rm1' 'rsvp1'
neighbor1 'rm1' 'qospf1'
neighbor1 'rm2' 'node2'
neighbor1 'rm2' 'rsvp2'
neighbor1 'rm2' 'qospf2'
neighbor1 'rm3' 'node3'
neighbor1 'rm3' 'rsvp3'
neighbor1 'rm3' 'qospf3'
neighbor1 'rm4' 'node4'
neighbor1 'rm4' 'rsvp4'
neighbor1 'rm4' 'qospf4'

peer 'realtime_node1-node3_so_1' 'realtime_node1-node3_si_1' (see Note 14)
peer 'realtime_node1-node3_so_2' 'realtime_node1-node3_si_2'
peer 'simple_node4-node3_so_1' 'simple_node4-node3_si_1'
```

Notes (Explanations):

1. component 'node1' NODE 0   0
   ----a---- --b-- -c— -d- -e-

a: ’component’ indicates that it is the beginning of a new compoent.

b: ’node1’ is the name of the component which is specified by the user. Any component must have a unique name.
c: ’NODE’ indicates the type of the component. The types of components are listed as follows.
NODE
LINK
REALTIME_SOURCE
REALTIME_SINK
RM
QOSPF
RSVP
SIMPLE_SOURCE
SIMPLE_SINK
TELNET_SOURCE
TELNET_SINK
FTP_SOURCE
FTP_SINK
PERF_MON
STOPPER
LINK_COST_FUNC

(d, e): (d,e) indicates the location of the component in the topology graph. It is not used in QRS.

2. param 'node1' 32   0
   --a-- --b-- -c- -d-
a: ’param’ indicates this is an input parameter.
b: ’node1’ indicates the input value of the parameter. The value is specified by user.
c: ’32’ specifies the flags of the parameter. The meaning of each bit is defined:
Bit 7: ModifyMask
Bit 6: LogMask
Bit 5: CanHaveLogMask
Bit 4: InputMask
Bit 3: CanHaveMeterMask
Bit 2: ChangeMask
Bit 1: DisplayMask
Bit 0: MeterMask
Bit 7 is the highest bit while Bit 0 is the lowest bit. If both Bit 6 and Bit 5 are set, the values of the parameter are logged into sim_log.ID. The value of ID specifies each instance of the simulation.
d: ’0’ specifies the Display type of the parameter. It is not used.

3. pflags 2e   4   # Buffer space used: 0
   ---a-- -b- -c-
a: ’pflags’ indicates it might not be an input parameter.
b: ’2e’ specifies the flags of the parameter.
c: ’4’ specifies the Display type of the parameter. It is not used.

4. In a configuration file, the parameters of each type of component must be configured in order. If the user changes the order of any two parameters, it might lead to unexpected results. Therefore, it is strongly suggested that the user configures the network configuration file through copying and modifying the example configuration file. Moreover, the user must not change ’pflags’ to ’param’ or vice versa.

5. In the component of LINK, the values of EWMA should be carefully specified. The ’EWMA weight’ must be set within (0, 1). The value of ’EWMA average limit’ might be set to a relatively large value (for example, 150000) when the user has no aim to study the effect of CBQ. I n this case, CBQ becomes PQ (priority queuing),

6. param 50000 82 0 # CBQ1: class A queue size : 50000
There are four types of classes.
Class A refers to the packets from RSVP and QOSPF.
Class B and C refer to the packets from Realtime Traffic.

Class D refers to the packets from best effort services, such as FTP, and Telnet.

7. param 1 82 0 # Flow index of this source : 1
Each pair of Realtime Traffic must have a unique interger within [0, 99].

8. param 1 82 0 # Class type of this source : 1
This parameter specifies the class of the Realtime Traffic component.

Value	Type
0	EF
1	AF1
2	AF2
3	AF3
4	AF4
5	BE

9. param 100000 82 0 # Time btw topology broad (msec): 1000
This parameter is neglected in the case that period based (PB) link state update method is used. In other cases, this parameter is used in case deadlock. It should be set to a relatively large value, e.g., 10000.

10. param 1000 82 0 # value of LS update period (*10us) : 1000
All instances of QOSPF must have the same value of the parameter. It should be neither too large nor too small.

11. param 0.1 82 0 # update by bandwidth percent variation: 0.1
All instances of QOSPF must have the same value of the parameter. It should be neither too large nor too small.

12. neighbor1 'node1' 'lk1-2'
---a----- --b--- ---c--
a: ’neighbor1’ indicates the first component is connected to the second compoennt.
b: ’node1’ is the first component.
c: ’lk1-2’ is the second component.
Therefore, the relationship of adjacency between two components is expressed as follows.
neighbor1 'node1' 'lk1-2'
neighbor1 'lk1-2' 'node1'

The following two types of component might be neighbors.
NODE – LINK
NODE – Realtime Traffic Source/Sink, Simple Traffic Source/Sink, FTP Source/Sink, Telenet Source/Sink
NODE – RSVP
NODE – RM
NODE – QOSPF
QOSPF – RSVP
QOSPF – RM
RSVP – Realtime Traffic Source/Sink
RSVP – RM

13. Each NODE component must be connected with one RSVP, one RM, one QOSPF, at least one LINK component.

14. peer 'realtime_node1-node3_so_1' 'realtime_node1-node3_si_1'
--a- -----------b-------------- ------------c------------
a: ’peer’ indicates the following two components are a pair of traffic.
b: 'realtime_node1-node3_so_1' is the Source of the traffic pair.
c: 'realtime_node1-node3_si_1' is the Sink of the traffic pair.

15. param 1 82 0 # CBQ1: AF WRED threshold 1: 0
The default value is 1. Note that WRED is not implemented in QRS v2.0 yet.

16. param 1 82 0 # CBQ1: AF WRED slop 1: 0
The default value is 1.

17. param 0 86 0 # Drop precedence of this source : 0

```
This parameter is only valid for AF class
Value  Drop precedence
0      0
1      1
2      2

18. param 0 86 0     # Resource reserve type of this source: WR: 0
Value  Type
0      With reservation
1      Without reservation

19. param 0 86 0     # Routing method for this source : 1
Value   Type
00      ONE_PATH_LOWEST_COST
01      ONE_PATH_WIDEST_BW
02      ONE_PATH_LOAD_SHARING
04      TWO_PATH_TE_LOWEST_COST
05      TWO_PATH_TE_WIDEST_BW
06      TWO_PATH_TV_LOWEST_COST
07      TWO_PATH_TV_WIDEST_BW
08      TWO_PATH_ED_LOWEST_COST
09      TWO_PATH_ED_WIDEST_BW
10      TWO_PATH_VD_LOWEST_COST
11      TWO_PATH_VD_WIDEST_BW

20. param 1 86 0     # The number of multipaths: 1 or 2
Value  Paths
1      1
2      2

21. param 0 82 0     # Load sharing method for this source: 0:
Value  Type
0      0

22. param 100 82 0 # Load sharing percent, e.g., 10 (percent): 0
If the number of multipaths is 1, the value should be 100. If the number of
multipaths is 2, the value w could be within [1,100]. The load sharing
percent of the other path is 100-w.
```

An incomplete or misconfigured file may lead to unexpected simulation results.

5.2 Running the simulation

Then, the user can simulate tree.cfg by typing (see section 2.3)

qrs –x -perfdt 1000 example-net/tree.cfg 100000

Each instance of simulation produces two files: sim_snap.ID and sim_log.ID. The former records the network configuration and the final value of parameters, for example,

```
pflags 2e 4    # Max buffer space used: 104
                                                -a-
a: ’104’ is the final value of the parameter at the end of the simulation.
```

The latter file records values of the parameters, whose bit 5 and bit 6 of PFLAGS are set in the configuration file. For example, in tree.cfg, two parameters are set to log, i.e.,

```
pflags 6e 4    # node1: Buffer space used
pflags 6e 4    # node1: Max buffer space used
```

The beginning of the sim_log.ID is

```
# Simulation started with seed 956757654 on ws2
# 1 'node1' 'Buffer space used'
# 2 'node1' 'Max buffer space used'
1 0 0
```

```
2 0 0
1 0 28
2 0 28
1 0 0
1 106 52
2 106 52
1 126 0
1 126 52

Note: The first value in each line is the number of the parameter logged;
The second value is the time; The third value is the value of the parameter
at that time. The unit of time is 10us.
```

5.3 Analyzing the network

The user can analyze the performance of the network through selecting the parameters and analyzing the log file.

Reference

1. *MaRS (Marland Routing Simulator) – Version 1.0 User's Manual*. C. Alaettinoglu, et al. June 1991. http://www.ccs.neu.edu/home/matta/publications.html
2. *Design and Implementation of A New Routing Simulator*. P. Zhang, R. Kantola, Z. Ma. Feb. 2000. http://www.tct.hut.fi/tutkimus/ironet/QRS/index.html
3. *MaRS (Marland Routing Simulator) – Version 1.0 Programmer's Manual*. C. Alaettinoglu, et al. June 1991. http://www.ccs.neu.edu/home/matta/publications.html
4. *Link-Sharing and Resource Management Models for Packet Networks.* S. Floyd, V. Jacobson. IEEE/ACM Transactions on Networking, Vol. 3, No. 4, Aug. 1995.